REMARKS

In response to the final Office Action mailed on August 2, 2005, Applicant respectfully requests reconsideration of all rejections in the outstanding Office Action in view of the foregoing amendments and following remarks. Claims 1-12 are currently pending.

I. Request For Interview

In the event that there are any issues left unresolved by this Reply, the Examiner is requested to contact the undersigned to schedule a telephone interview prior to issuance of another Office Action. The undersigned can be reached at the number listed below.

II. § 101 Rejection Of Claims 5-12

Claims 5-12 stand rejected under 35 U.S.C. § 101, as allegedly directed to unpatentable subject matter. *See* Office Action, August 2, 2005, page 3. Based on the assumption that these claims are limited by the method they perform, the Examiner contends that these claims are unpatentable because: (i) claims 5-8 manipulate only numbers and do not require the use of any tangible hardware, (ii) human mental processes would infringe means-plus-function claims 9 and 10, and (iii) system claims 11 and 12 structurally refer only to processors, which are not integral to or contribute functionality to the claimed system. *See id.* Applicant respectfully disagrees and traverses this rejection on the following grounds.

A. Process Claims 5-8 Are Statutory Because They Recite A Practical Application

The Manual of Patent Examination Procedure, 8th Edition, Revision 2 (2004), articulates that a computer-related process should be considered statutory (i.e., patentable subject matter) if the process is limited to a practical application in the technological arts.

> For such subject matter to be statutory, the claimed process must be limited to a practical application of the abstract idea or mathematical algorithm in the technological arts. MPEP § 2106 (IV)(B)(2)(b)(ii) (citations omitted).

Likewise, a process that consists solely of mathematical operations without some claimed practical application is considered nonstatutory.

> In practical terms, claims define nonstatutory processes if they:
>
> -- consist solely of mathematical operations without some claimed practical application (i.e., executing a "mathematical algorithm"; . . . . MPEP § 2106 (IV)(B)(1)

Claims 5-8 specifically recite a practical application. Independent claim 5 claims a "process for compiling or translating a computer program code instruction using transformed address space." In the computer arts, compiling or translating a computer program code is a "practical application" that enables software in one format to run on hardware implemented a different software format. Claims 6-8 depend from claim 5. Accordingly, claims 5-8 define statutory subject matter as they are limited to a practical application, i.e., compiling or translating a computer program code, and not merely an algorithm without a practical application.

B. The Means-Plus-Function Element in Claims 9 And 10 Must Be Interpreted To Read On Only The Structure Disclosed In The Specification And "Equivalents Thereof."

Claims 9 and 10 are statutory because they recite subject matter that falls within the four categories of invention that Congress deemed to be the appropriate subject matter of a patent, namely, processes, machines, manufactures, and compositions. *See* MPEP § 2106 (IV)(A) (*discussing* 35 U.S.C. § 101). Claims 9 and 10 are clearly directed to a system, which is a machine and/or manufacture explicitly contemplated as patentable subject matter under 35 U.S.C. § 101.

The Examiner's basis for rejecting claims 9 and 10 under 35 U.S.C. § 101 is unsound. Particularly, the Examiner contends that "[c]laims 9 and 10 are directed to a system defined in means-plus-function language and therefore must be broadly interpreted as similar in scope to the process of claim 5." *See* Office Action at page 4. This contention is simply wrong. Although claims 9 and 10 do recite means plus function language, such language must be interpreted to read on only the structures disclosed in the specification. Referring to MPEP § 2106 (II)(C):

> Where means plus function language is used to define the characteristics of a machine or manufacture invention, claim limitations must be interpreted to read on only the structures or materials disclosed in the specification and "equivalents thereof." (citations omitted) (emphasis added)

The use of means plus function language in claims 9 and 10 thus defines the claimed invention by the structure (e.g., processor(s) and/or memory) disclosed throughout the specification rather than a process recited in another claim. The Examiner's interpretation of these claims is inconsistent with the specification and contrary to well-settled patent law.

C. Claims 11 and 12 Are Limited To A System Comprising A Memory And A Processor

Similarly, claims 11 and 12 are statutory because they are clearly directed to a system, which is a machine and/or manufacture explicitly noted as patentable subject matter under 35 U.S.C. § 101. *See* Remarks § I.B. Moreover, these systems are expressly limited to a system "comprising: a memory . . . and a second type of processor . . ." and accordingly have structure. The Examiner is not permitted to simply disregard structural recitations and conclude that the claims are limited "by the method they perform." *See* Office Action at page 4. Applicant can find no such precedent for permitting such a conclusion.

Based on the above reasoning, Applicant respectfully submits that this § 101 rejection is unsustainable.

III. Indefiniteness Rejection Of Claims 1-4

Claims 1-4 stand rejected under 35 U.S.C. § 112, ¶2, as allegedly being indefinite for failing to point out and distinctly claim the subject matter which applicant regards as the invention. *See* Office Action at page 14. Particularly, the Examiner objects to the use of the terminology "emulating a processor." Although Applicant respectfully disagrees, claims 1-4 have been amended to recite a "computer-implemented method for use in emulating a processor . . . ." This revised claim language makes clear that the recited method itself does not perform emulation, and correctly states that the method is suitable for use in emulation. Applicant respectfully submits that this § 112 rejection is unsustainable in view of these amendments.

IV. Anticipation Rejection Of Claims 1-13

Claims 1-13 stand rejected under 35 U.S.C. §102(b), as allegedly anticipated by U.S. Patent No. 5,968,164 to Loen et al. ("Loen"). *See* Office Action, page 7. Particularly, the Examiner contends that Loen teaches each and every element recited in claims 1-13. Although Applicant respectfully disagrees, the pending claims have been amended to better distinguish the claimed invention over the prior art. Support for these amendments is found at least at page 1, paragraph 3 and page 3, paragraph 4.

Applicant respectfully submits that the rejection is unsustainable in view of these amendments and the following remarks.

As stated in MPEP 2131, "A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art

reference." *Verdegaal Bros. v. Union Oil Co. of California*, 2 USPQ2d 1051, 1053 (Fed Cir. 1987).

A. Overview Of Loen Vis-À-Vis Claimed Invention

Using Loen's method, the inner loops of each emulated software program repeatedly read and transform the same data words. *See* Loen, col. 3, line 12 *et seq.* ("It is a principal object of this invention to provide an enhanced computer system that supports tasks of different endian without the need for reinitialization."). Loen is an example of the prior art, which exhibits exactly the problem which is addressed by the present invention. *See* Applicant's Specification, page 1, paragraph 3 ("Generally, systems of this type convert each word between endian transformations on a word by word basis. This conversion, when required frequently, introduces a significant overhead into the time required to perform a given task"), and paragraph 4 ("It is an object of this invention to provide an efficient method and system to enable software of one endian format to run on hardware of a different format."). Accordingly, in the present invention, the inner loops of each emulated software program repeatedly read an instruction, but are not required to transform memory access address of the instruction before executing, emulating, or processing the instruction.

According to Loen, the most frequently accessed data words are repeatedly read and transformed, and transformed yet again if the words are changed and stored back into the memory address space. *See* Loen, col. 3, line 53 *et seq.* ("The mixed-endian mechanisms automatically format the data in the form expected by the running task, regardless of whether the task expects the data to be in big endian format or in little endian format."). In the claimed invention, multiple data words are transformed together once, before emulation is performed, and optionally a second time after emulation has been completed

The execution time of an emulation is a very important attribute in the emulation process. Loen's approach is more computation-intensive compared to that of the Applicant's. Loen's method specifies transformation of every memory access to present the proper endian format to the processing unit or processing program employed to emulate a processor of a different endian format. Transformation is needed for every read and every write of either instruction words or data words in Loen's method.

By contrast, the claimed invention supports a single transformation of a plurality of words prior to the process of emulation. Subsequently, when the processor or software is emulating a processor of a different endian type, it is no longer necessary to transform each word read or written to a memory addresses. Finally, after the emulation process has been completed and the emulation process or software program halts, an optional second transformation of each of the plurality of words in the memory restores the original endian format. Using the present invention, the number of transformations has been reduced by a large, application-dependent fraction as compared to Loen's method.

B. Loen Only Teaches Reflection Of A "Data Double Word Or Fragment Thereof"

As discussed previously, Loen discloses a two-step process where:

> The first step is a reflection which must be performed on the bytes comprising the data double word or fragment thereof (see Fig. 3a). The second step is a modification of the memory address offset of the bytes comprising the data double word to accommodate the new location of the bytes after the reflection that was performed in the first step (see Fig. 3b). [*see* Loen, Column 7 lines 5-11]

As identified by the Examiner, Loen then teaches that:

> The reflection step may be performed in a variety of places and is entirely mechanical and unrelated to the data element size being fetched [column 7, lines 12-14].

However, this sentence at column 7, lines 12-14 cannot be read in isolation and cannot be divorced from the clear statement of the immediately preceding lines. Instead, lines 12-14 must be read together with lines 5-11. Here, the reflection step "must be performed on the bytes comprising a data double word or fragment thereof." There is no teaching in Loen that the reflection step is performed on anything other than a "data double word or fragment thereof".

As noted above, the reflection step of Loen is only the first step. Loen also teaches the second step, namely "modification of the memory address offset of the bytes comprising the data double word to accommodate the new location of the bytes." That is, Loen's teaching of the second step reinforces that the reflection and address modification steps are performed individually on each data double word or fragment thereof. In particular, lines 12-14 must be compared with lines 35-36 of Loen discussing the address modification step:

> The second step performs an address modification which depends on the size of the data word which is being referenced by the processor [column 7, lines 35-37]

Lines 12-14 refer to size purely as a contrast to lines 35-37, and must be read in that context.

C. Loen Uses XOR To Modify The Lower Order Address Bits

Applicant's interpretation of Loen is further reinforced by the manner in which the address modification is performed. As discussed at column 7 line 35 to column 8 line 46 of Loen, the address of individual bytes in each data double word or fragment thereof are modified by an XOR of the lowest bytes of the address. In the described example of Figure 3b, the address offset modification is generalized for the lowest three bits of the address. The address modification step can only operate on a reflection of a data double word or fragment thereof.

D. The Transformation Of Loen Is Not Equivalent To The Claimed Invention

It is important to recognize that, even if the reflection and address modification steps of Loen are performed repeatedly on a series of data double words in memory, the resulting transformation is not equivalent to the memory address transformation as performed by the claimed invention.

The transformation mechanism of Loen gives rise to exactly the problems identified in the present application at paragraph 4. Namely, the transformation of Loen converts each word (double word or fragment thereof) between endian representations on a word by word basis (double word by double word/fragment by fragment). This conversion, when required frequently, introduces significant overhead into the time required to perform a given task.

It is helpful to illustrate this fundamental difference with the following example. For convenience, the example is drawn based on Figures 3 and 4 of Loen, when expanded to show three successive data double words in memory.

Fig. 1 shows three data double words arranged sequentially in memory in little endian format within a little endian address space.

Word C	C1	C2	C3	C4	C5	C6	C7	C8
Address	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*
Word B	B1	B2	B3	B4	B5	B6	B7	B8
Address	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
Word A	A1	A2	A3	A4	A5	A7	A7	A8
Address	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*

Fig 1

Figure 2 shows the same three data double words when reflected individually as taught by Loen (i.e. following the double word reflection shown in Figure 3A of Loen). Effectively, there is a reflection of the bytes within each word about a vertical plane, as shown by the dotted line between columns C4 and C5.

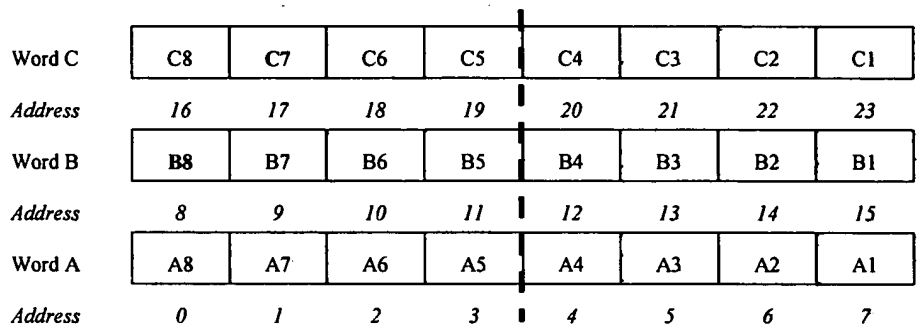


Fig 2

By contrast, Figure 3 shows address transformation according to the present invention. Effectively, there is a reflection of bytes in both vertical and horizontal planes about the two dotted lines shown.

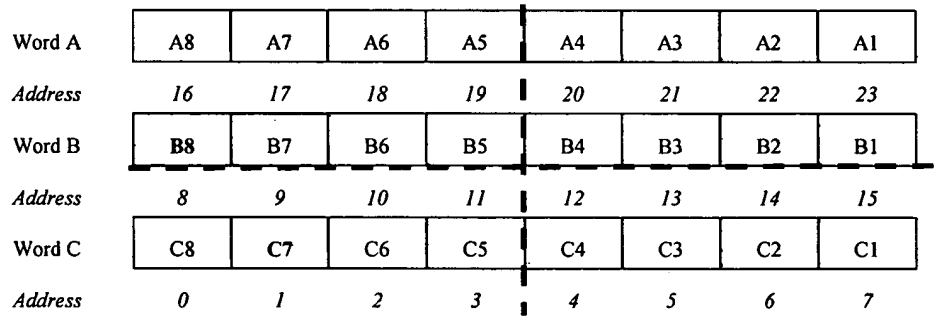


Fig 3

E. Claims 1 and 9

Loen fails to teach each and every element recited in independent claims 1 and 9.

The Examiner cites definitions of "address space" which are all equally applicable to the claimed invention. However, the claim language has been clarified by instead referring to "memory access addresses of a plurality of words," namely:

> transforming memory access addresses of a plurality of words such that bytes stored in a memory addressed by a processor of the second type as a result of an instruction in which a byte order in accordance with the first convention is observed are distributed in a pattern which is a mirror image of the distribution

> pattern of the bytes which would result if the memory was addressed by a processor of the first type in response to the said instruction. [Emphasis added.]

As discussed above, the reflection of Loen is performed on "a data double word or fragment thereof." Loen does not teach transforming memory access addresses of a plurality of words, as in claim 1 and claim 9.

F. Claims 2 and 10

Loen fails to teach each and every element recited in independent claims 2 and 10.

The transformation of the present invention is described in detail at paragraphs 27-45 of the present application and fulfils the conditions recited in claim 2, namely that:

> (a) an offset between addresses of any two bytes stored in the memory is unaltered by the transformation, where said any two bytes are spaced apart in memory by a plurality of words; and
>
> (b) the relative order of the addresses of said any two bytes stored in the memory is reversed by the transformation.

The Examiner notes that conditions (a) and (b) are met in Loen for bytes within a single data double word. However, the conditions (a) and (b) of claim 2 are not satisfied by the teachings of Loen for any two bytes in memory. In particular, Loen does not satisfy conditions (a) and (b) of claim 2 for bytes which are spaced apart by a plurality of words and are not within a single data double word, i.e. between a byte in a first data double word and a byte in a second data double word.

It follows that condition (a) is not satisfied by Loen because the offset between any two bytes is altered by the transformation of Loen. Take, for example, bytes C7 and B8. In Fig 1, the offset between byte C7 and byte B8 of Fig. 1 is 22 - 15 = 7 bytes. Also, in Fig 3, the offset between byte C7 and byte B8 of Fig. 3 is 1 - 8 = 7 bytes. By contrast, in Fig 2 (Loen) the offset between byte C7 and byte B8 of Fig. 2 is 17 - 8 = **9 bytes**.

That is, in Figure 2 which represents Loen, condition (a) of claim 2 only holds true for some of the bytes, such as for byte B1 and C1. However, condition (a) of claim 2 holds true for every two bytes of Figure 3, representing the present invention.

Similarly, condition (b) of claim 2 is not met by Loen for any two bytes. In Fig 1, the relative order is that byte C7 (22) has a higher address than B8 (15). In Fig 3, representing the present invention, the relative order is reversed so that C7 (1) now has a lower address than B8

(8). By contrast in Fig. 2, representing Loen, the byte C7 (17) still has a higher address than byte B8 (8) such that the relative order of these two bytes is clearly not reversed.

In many practical situations, a program stores a block of data in a large plurality of words in memory. Often, the program will employ relative addressing such that a register holds a start address of the block, and a variable offset is incremented to selectively access a set of bytes in memory. As shown in the highlighted example, constant offsets are maintained between any two bytes spaced apart in the memory and the offset is not altered by the transformation. As a result, the claimed memory address transformation allows a target program on a target processor to employ a register + variable offset addressing mechanism which is closely equivalent to the subject program of a different-endian subject processor.

In summary, there is a fundamental difference between the memory transformation of the claimed method and the transformation disclosed by the prior art of Loen. Claims 2 and 10 precisely recite those differences.

G. Claims 3 and 11

Loen fails to teach each and every element recited in independent claims 3 and 11. Specifically, claim 3 recites:

> "transforming memory access addresses such that strings of bytes in the first endian format in a plurality of words which are stored successively by the processor... aggregate [as if] the processor was of the first endian format and memory access addresses were not transformed."

Figure 3A of Loen discloses that a string of bytes are stored successively, but only within one data double word or fragment thereof. Loen performs conversion on a word by word basis. Loen does not fulfill the condition that, for a plurality of words, strings of bytes in the first endian format aggregate as recited.

Again, this can be illustrated using the above example where Figure 3 representing the present invention aggregates the strings of bytes both within words and within adjacent words, whereas Loen aggregates strings of bytes only within words and does not aggregate strings of bytes in a plurality of words.

For example, let us examine bytes B8 through to C7. In Fig 1, the string of bytes B8 to C7 successively aggregate in addresses 15 to 22. In Fig 3 (representing the present invention), the string of bytes B8 to C7 successively aggregate in addresses 8 to 1. Even though the bytes are

stored successively by (in this case) a big endian processor, aggregation is not disturbed. By contrast, in Fig 2 (Loen) there is a dislocation at the word boundary between byte B8 (address 8) and byte C1 (address 23), and the string of bytes B8 to C7 does not aggregate successively.

H. Claims 4 and 12

Concerning claims 4 and 12, the claimed method recites transforming each of a plurality of memory access addresses using the formula "A - B - L + S", namely:

> transforming a plurality of memory access addresses relating to a plurality of words such that each memory access address B of string length L is transformed to the address A-B-L+S, wherein A is the total number of bytes allocated to a program, and S is the start address of the program

The two examples cited by the Examiner referring to Figures 4A and 4B of Loen require an improper interpretation of the claim language.

Concerning the Figure 4A example, the condition recited in claim 4 is only achieved by Loen when the total memory allocated is A equals 8 bytes and the start address of the program is S equals 0 bytes. The transformation fails if the total memory allocated is any greater than 8 bytes, or where the program does not start at address 0. Any real program in a practical computer system operates on a far greater memory allocation than 8 bytes, and would not start at address 0.

Claim 4 recites transforming a plurality of memory access addresses relating to a plurality of words. Loen only transforms data word by word (i.e. a data double word or fragment thereof).

An advantage of the invention lies in transforming a plurality of memory access addresses prior to run time, such that the transformed memory addresses are available to run time version of the program. Although the present invention incurs additional overhead at start up, there is a significant saving at run time and a substantial benefit overall, in contrast to the word by word transformation taught by Loen. In use, factors such as the string length L, the total program allocation A, and the start address of the program S, are known and will generally remain constant for a significant period of operation. Hence, the only variable in the equation A-B-L+S is now the memory access address B. In use, this memory access address B is readily transformed with a single subtraction operation X - B, where X = A - L + S. Hence, the transformation method as claimed significantly reduces overheads during run time.

I. Claims 5-8

Claim 5 is clearly distinguished over the prior art of Loen. As shown above in Figures 1-3, Loen does not perform memory address transformation:

> with respect to a fixed block size of a plurality of words in memory in the predetermined programmable machine so as to change the referenced address value by an amount that is fixed for a given number of bytes being accessed in each word

Secondly, Loen does not:

> [include] the thus changed address reference in a compiled or translated output instruction so that there is no extra operation required during execution of the output instruction to accommodate the convention for ordering bytes within words used by said predetermined programmable machine.

Dependent claims 6-8 are allowable at least because they depend from claim 5.

Claim 13 has been cancelled.

In sum, because Loen fails to disclose each and every element recited in the independent claims, the rejection of claims 1-5 and 9-12, and any claims dependent therefrom, i.e., claims 6-8, is unsustainable. Accordingly, Applicant respectfully requests the Examiner to withdraw this rejection.

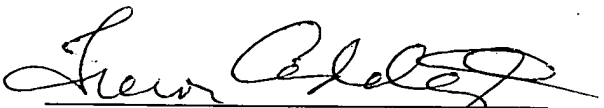
V. Conclusion

In view of the foregoing, it is respectfully submitted that the present application is in condition for allowance, and an early indication of the same is courteously solicited. Applicant is concurrently submitting herewith a Request for Continued Examination (RCE) along with the requisite fee. In the event that a variance exists between those fees necessary for the entry and

consideration of this reply or to maintain the instant application pending, the Commissioner is hereby authorized to charge or credit such variance to the undersigned's Deposit Account No. 50-2613 (Order No. 45256.00004.CIP1.P1068).

Respectfully submitted,

November 2, 2005

By: 

Trevor Q. Coddington, Ph.D., Esq.
Registration No. 46,633

PAUL, HASTINGS, JANOFSKY & WALKER LLP
Customer Number: 36183
P.O. Box 919092
San Diego, CA 92191-9092
Telephone: (858) 720-2500
Facsimile: (858) 720-2555